

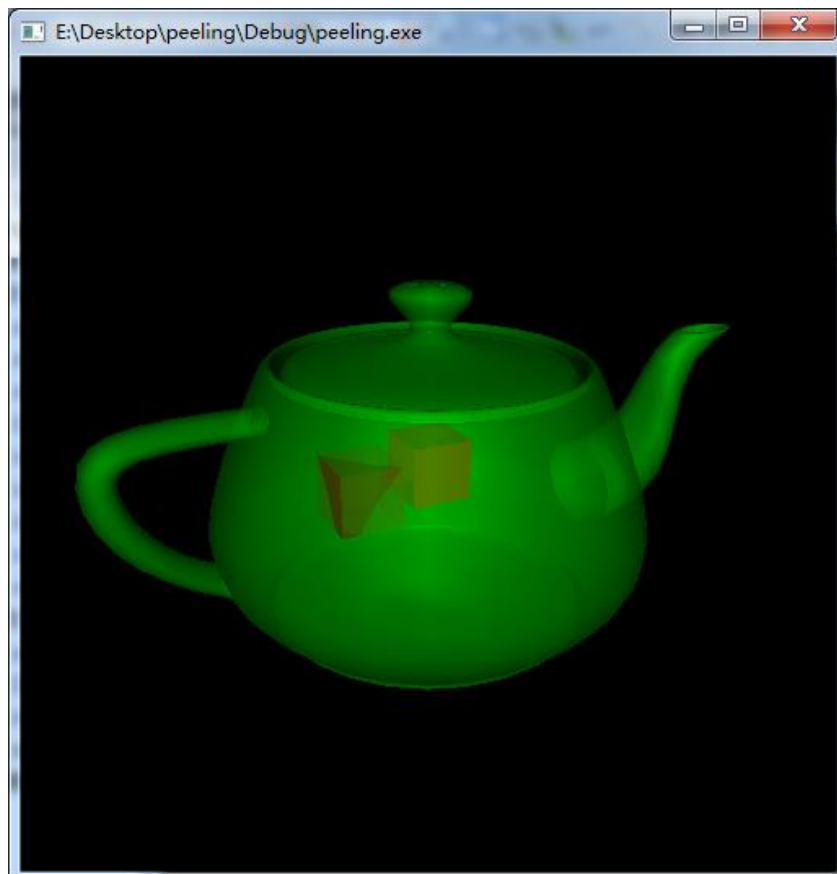
# 本周周报（5.6-5.12）

刘昊南

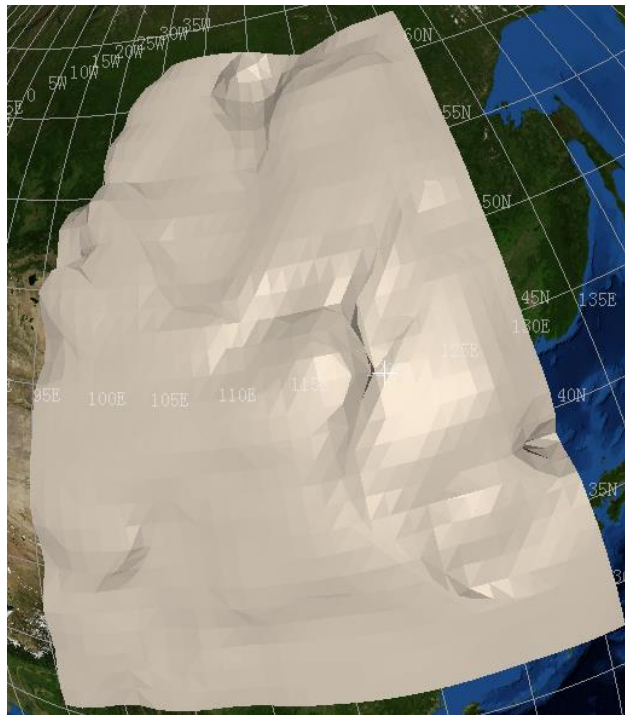
## 本周工作

1. 解决了上周发现的 `depth peeling` 无法剥离不垂直于视线方向的面 bug。起初认为是浮点精度问题，OpenGL 的默认深度缓冲是 24 位浮点，而 `shader` 中计算的深度是 32 位浮点。但是在采用 32 位的深度缓冲后，bug 并没有消除。后来发现，在设定纹理的过滤模式时，采用的是 `GL_LINEAR` 模式，即在取深度纹理的值时会对靠近纹理坐标的一块 `2x2` 纹理单元取加权平均，即进行双线性插值，从而得到的是不准确的深度值。由于深度纹理与场景的分辨率是一样的，即像素和纹理单元是一一对应的，我们对在每一个像素位置上做 `depth peeling` 时，需要取得的是纹理中同样位置的那个纹理单元的值，而不是插值后的值，所以应该将纹理的过滤模式设为 `GL_NEAREST`，即取最靠近纹理坐标的纹理单元的值，这个 bug 便解决了。

下面是加入了双面光照的一个 demo，茶壶内部的小立方体是不透明的，不完全在茶壶内部的小立方体是半透明的



2. 解决了等值面的渲染中光照效果的问题，之前由于等值面的法线计算不对，导致起伏不平的效果出不来。目前每个顶点的法线取的是三角形面片的法线，并不是相邻三角形法线的平均，所以不是那么平滑。下图为温度场等值面



3. 和海东师兄讨论了如何在项目中嵌入 **depth peeling** 功能。目前有个问题需要解决，**shader** 的合并问题。目前的项目代码中，体绘制、等值面、球面等用了不同的 **shader**，而其他的绘制对象则用的是固定管线，在绘制不同的物体时，需要在不同的 **shader** 和固定管线之间切换。由于 **depth peeling** 需要在 **fragment shader** 中做的，因此，有必要将所有固定管线改为使用 **shader**，修改所有的 **fragment shader** 加入 **depth peeling** 功能，这样工作量就很大了，而且我要找到所有绘制的地方，修改代码。

经过讨论，采取了一个折中的办法，即仅对等值面、切片这两种半透明物体渲染时进行 **depth peeling**，而其他的不透明物体不采取 **depth peeling**。半透明物体和不透明物体各采用一个 **fbo** 进行绘制。最后将半透明物体剥离的几层纹理，和不透明物体绘制结果进行混合。这样做的优点是，我仅需要修改体绘制和等值面的 **shader**，并在切片的绘制中加入 **shader**，而这几部分的代码我也比较熟。缺点是在最后的进行混合的 **fragment shader** 中，除了需要输入所有的颜色纹理，还需要输入所有的深度纹理。因为半透明物体的几层纹理是深度剥离出来的，是按由近到远的深度顺序的，但是不透明物体的纹理与它们没有这样的深度上的关联，也许在某几个像素上深度比半透明物体的几层纹理都浅，在某几个像素上深度比半透明物体的几层纹理都深。因此，在混合时需要深度纹理作为参考。

## 下周计划

1. 实现没有体绘制时的混合
2. 实现加入体绘制的混合